ENPM673– Spring 2022

Sumedh Koppula[1]
[1]University of Maryland, College Park, Maryland- USA
e-mail: sumedhrk@umd.edu, UID: 117386066

Problem 1: Histogram equalization

Here, we aim to improve the quality of the image sequence provided above. This is a video recording of a highway during night. Most of the Computer Vision pipelines for lane detection or other self-driving tasks require good lighting conditions and color information for detecting good features.

A lot of pre-processing is required in such scenarios where lighting conditions are poor. Now, using the techniques taught in class your aim is to enhance the contrast and improve the visual appearance of the video sequence.

You cannot use any in-built functions for the same. You have to use histogram equalization based methods, both histogram equalization and adaptive histogram equalization and compare the results.

Solution

Simple Histogram Equalization

Steps followed:

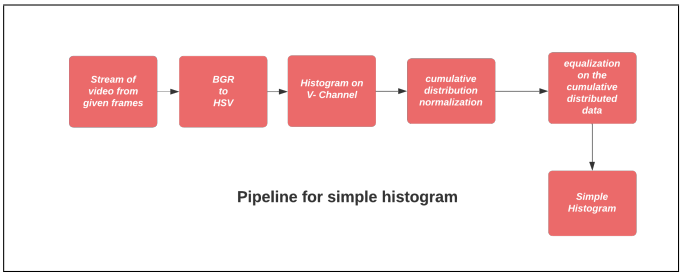

Fig. 1. Pipeline for simple histogram

1. From the given data consisting of 25 images, I had converted all the frames into a stream of video.

2. I have applied BGR2HSV on each frame

3. From the obtained HSV, I have extracted V channel and applied Histogram on it.

Fig. 2. Input Frames

4. Now, from the obtained Histogram list, I had calculated the cumulative distribution of the histogram data by normalizing N pixels with intensities less than or equal to i intensities.

5. Then I have performed equalization on the cumulative distributed histogram data. The final histogram is shown in the figure 3.

Fig. 3. Histogram output

Fig. 4. Comparison between Input frames and Histogram applied frame

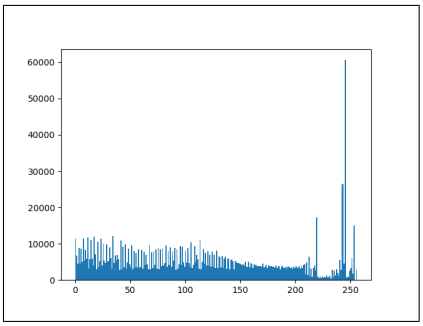

Fig. 5. Histogram Plot

Adaptive Histogram Equalization

1. From the given data consisting of 25 images, I had converted all the frames into a stream of video.

2. split each frame into 5*5 grid.

3. Applying histogram on each and every individual title of

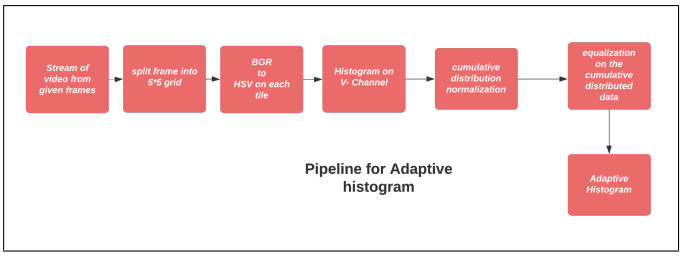

Fig. 6. Pipeline for Adaptive histogram

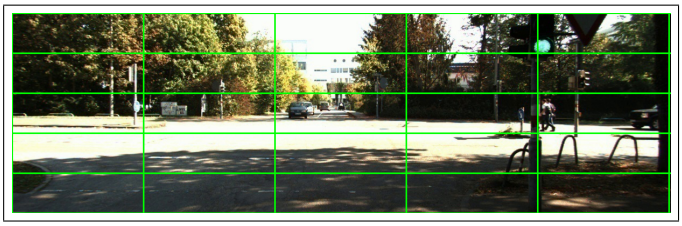

Fig. 7. Frame Split into tiles

the frame.

4. I have applied BGR2HSV on each tile

5. From the obtained HSV, I have extracted V channel and applied Histogram on each title.

6. Now, from the obtained Histogram list, I had calculated the cumulative distribution of the histogram data by normalizing N pixels with intensities less than or equal to i intensities.

7. Then I have performed equalization on the cumulative distributed histogram data. The final Adaptive histogram is shown in the figure 8.

8. Finally, I have compared input frame with obtained simple histogram and adaptive histogram.

Fig. 8. Adaptive Histogram Frame

Fig. 9. Comparison between Input frames, Simple Histogram, Adaptive Histogram

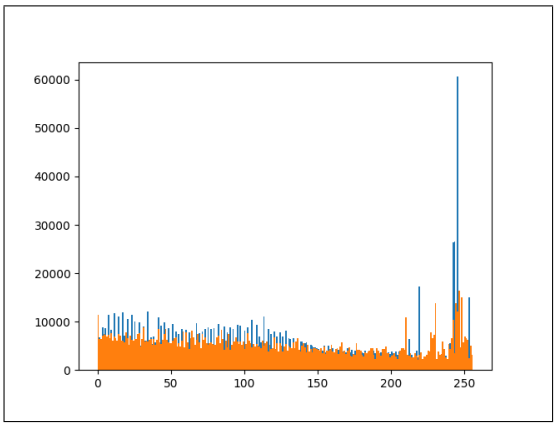

Fig. 10. Adaptive Histogram Plot

Problem 2: Straight Lane Detection

In this problem we aim to do simple Lane Detection to mimic Lane Departure Warning systems used in Self Driving Cars. You are provided with a video sequence, taken from a car.

Your task is to design an algorithm to detect lanes on the road, and classify them as dashed and solid lines. For classification of the line type, you have to use different colors. Use green for solid and red for dashed.

Solution

1. Initially, From for the given whiteline.mp4, each frame is applied to an edge detector.

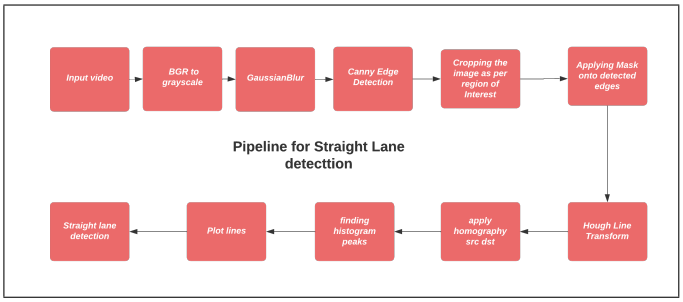

Fig. 11. Pipeline for Straight Lane Detection

2. Each frame is converted from BGR to grayscale

3. A Gaussian kernel is used. It is done with the function, cv.GaussianBlur(). I specified the width and height of the kernel which should be positive and odd. Gaussian blurring is highly effective in removing Gaussian noise from an image.

4. Applying Canny Edge Detection onto filtered frames. First argument is our input image. Second and third arguments are our minVal and maxVal respectively. Fourth argument is aperturesize. It is the size of Sobel kernel used for find image gradients. By default it is 3. Last argument is L2gradient which specifies the equation for finding gradient magnitude. If it is True, it uses the equation mentioned above which is more accurate, otherwise it uses this function: EdgeGradient(G)=|Gx|+|Gy|.

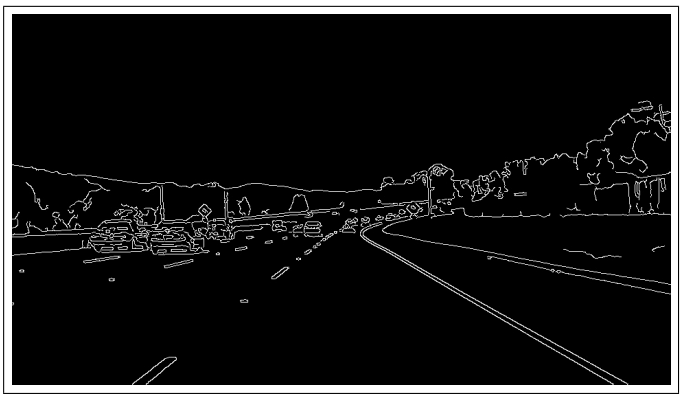

Fig. 12. Canny Edge Detection

5. Cropping the image as per our region of interest.

6. Applying Mask onto detected edges using bitwiseand()

7. In order to detect lane line. I am using a more efficient implementation of the Hough Line Transform. It gives as output the extremes of the detected lines (x0,y0,x1,y1)

8. In this step, I had applied homography between source and destination points using RANSAC method.

9. From the obtained homography, I had warped the masked edge frame as shown in the Figure 13.

10. Evaluated left lane and right lane by **finding histogram peaks** of the warped detected edges.

11. Depending on the left lane and right lane count, I had drawed the lines. If a white straight line is detected then I had drawed the line with Bold green color. If the a dotted white line is detected then i have ploted the lines with Red color.

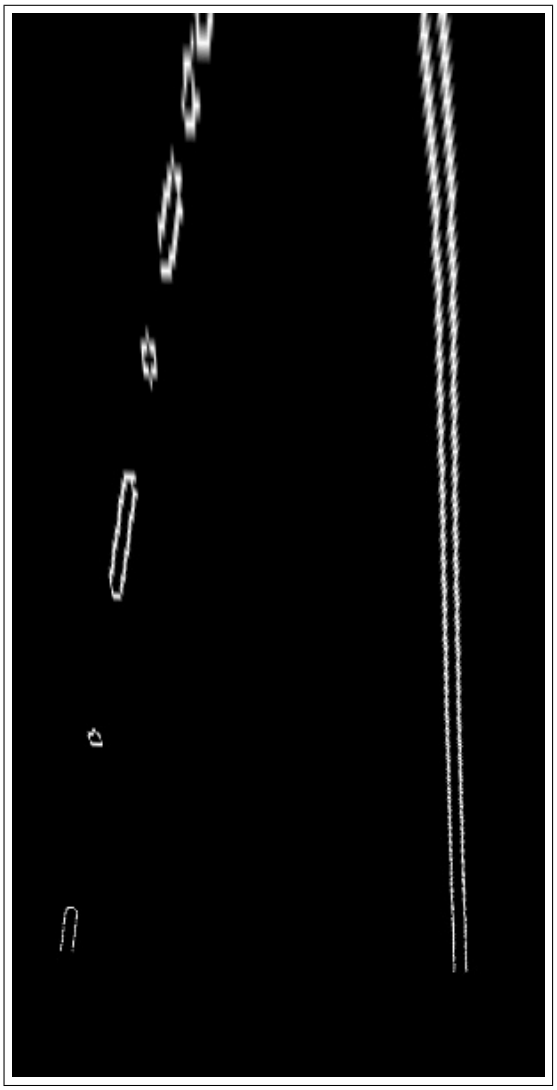

Fig. 13. Warped frame

Fig. 14. Straight lane detection

Problem 3: Predict Turn

In this problem, we aim to detect the curved lanes and predict the turn depending on the curvature: either left or right turn. The dataset provided has a yellow line and a while line. Your task is to design an algorithm to detect these lanes on the road, and predict the turn. Please note that for the output of the lane detection, you have to superimpose the detected lane as shown in fig.4. Also compute the radius is curvature for the lane using the obtained polynomial equation. Refer to this. When your lane detection system loses track(cannot find lane

lines), you must manage to use the past history of the lanes from the previous image frames to extrapolate the lines for the current frame.

Solution

1. From the given input video stream, I had found undistorted frame using the given camera calibration matrix (K) and distortion coefficients D.

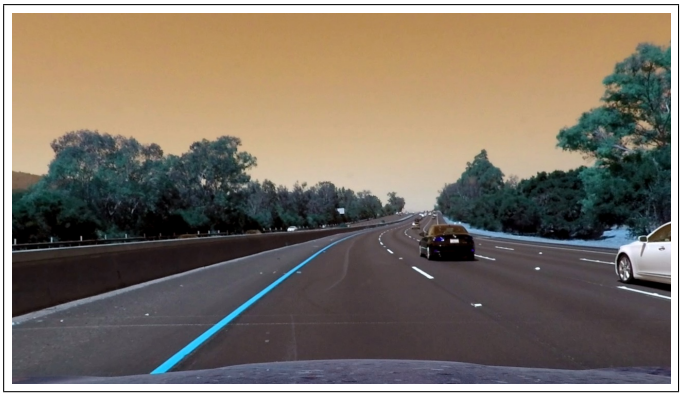

Fig. 15. Undistort Frame

2. Now I chose to find the lanes using combined color and gradient threshold

3. To enhance the image, Adaptive Histogram with Limited Contrast Equalization is used to adjust lane line visibility in varied lighting situations and in shadows.

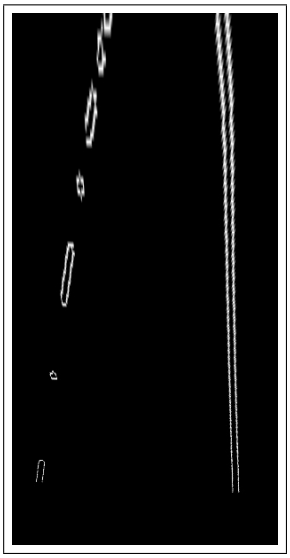

Fig. 16. Warped Lanes (Bird eye view)

4. converting the image from RGB to LAB color. Then appling CLAHE to the L channel.

5. I had cropped the processed frame as per the region of interest. Then applied gradient thresholding to perform perspective transformation

6. lanes are extracted using RGB and LAB color space.// The RGB channel aids in the isolation of white pixels in the majority of lighting settings, whereas the LAB channel aids in the isolation of yellow pixels in the majority of lighting conditions.

7. The above thresholded white and yellow color masks are then blended as a single mask using the OR condition.

8. Two methods for gradient detection are employed and coupled to try to acquire more sources of information for robustness.

Method 1: Applied a denoise filter and a gradient threshold along the X-axis to a warped image.

Method 2: On the original road image, apply gradient thresholds by magnitude and direction independently, combine them with an AND condition, apply a denoise filter, and warp to perspective.
Methods 1 and 2 data are then combined with an OR condition to overlap the observed lane lines from both sources.

9. Gradient detection is influenced by noise and can be difficult to extract lane lines for varied settings, hence employing numerous algorithms may help extract lanes for different conditions.

10. If no preceding lanes are recognized, we apply both color and gradient masks for a more restrictive selection of lane pixels, reducing the number of missed detections due to noise. If a lane was detected in the previous frame, we just employ color mask for a more broad selection because the search is already constrained to a window surrounding the known lane location.

11. **Slidding window based histogram** and Determining valid Lanes:

I started at the bottom one-fourth of the image, with windows centered at beginning x coordinates determined by two maximum peaks in vertical slice histograms.

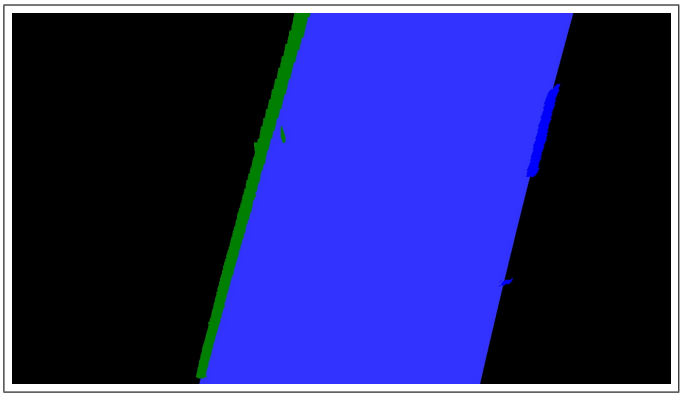

Fig. 17. Lanes detected

To set the center of the next window, I utilized the average x position of the points within the window. If not enough points are found on one side but points are found on the other side, change the window center based on the previously identified lane width between left/right windows.

If there aren't enough points on either side, keep sliding the window centers by the shifts calculated by the previous window.

To reduce noise, narrow the window for the final lane x,y points and appling a 2nd order polyfit to the final lane x,y

points.

In the lane objects, I saved the detected lane x,y values and polyfits.

The technique is even easier if good lanes have been spotted in prior frames. Within a defined margin width of the existing polyfit lane lines, we select any lane x,y points. Then, on the final lane x,y points, perform a 2nd order polyfit. In the lane objects, save the detected lane x,y values and polyfits.

12. Acceptable lane width conditions were utilized to overcome the drawbacks of the thresholding procedure, while acceptable lane pixel conditions were used to overcome noise from the detecting process.

13. We get pixels to meters conversions based on our distorted image dimensions after we have the lanes and their related locations.

14. **Radius of curvature and lane center offset calculation**

$$R = \frac{(1+(2A+B)^2)^{\frac{2}{3}}}{2A} \tag{1}$$

The lane center offset is derived by assuming the automobile is at the image's midpoint and measuring the distance between each lane line and the car's center.

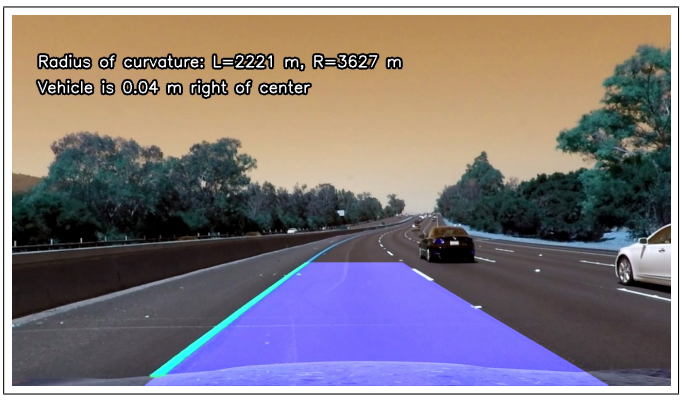

Fig. 18. Road Lanes detected

15. Return the lane detected image to the original road image by unwarping it. Finnally, Combining all the intermediate output frames into a single frame for displaying output effectively

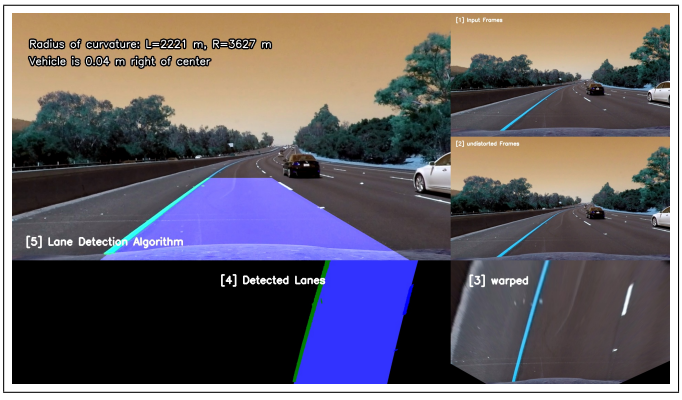

Fig. 19. Combined output Lanes detection algorithm